CHAPTER

6

DATA STORAGE

After reading this chapter and completing the exercises, you will be able to:

- Explain how physical disks are organized
- Describe how Windows 2000 reads disks to circumvent the logical limit to partition size
- Explain how the physical structure of a hard disk relates to its logical organization
- List the types of volumes supported by Windows 2000
- Discuss the role of a file system in an operating system
- Describe the features of each file system supported by Windows 2000
- Examine the advanced features of NTFS

One of the most important parts of any operating system is its file storage mechanisms. Good file storage provides security from intruders and prevents data loss; bad file storage leaves your organization's data vulnerable to loss. This chapter explains how the Windows 2000 file storage works and discusses the use of volumes and file systems in Windows 2000.

UNDERSTANDING DISK ARCHITECTURE

Let's first consider how hard disks are organized independently of Windows 2000 and how that physical organization relates to the logical storage units. We begin this discussion with an exploration of physical disk structures, then take a look at logical disk structures.

Physical Disk Divisions and Important Files

A hard disk—the metal and plastic contraption inside the computer on which you store data—consists of a set of round, magnetized, metal **platters** organized into a stack. Each side of each platter, called a **surface**, is divided into concentric circles called **tracks** located parallel to one another on each surface of each platter. An electromagnetic read/write **head**, like the needle on a record player, reads the surface of the disk. All heads in the disk move in unison so that they always point to the same track. That is, if head 0 (the first head on the first platter in the disk) is positioned over track 78 on surface 0, then head 4 is positioned over track 78 on surface 4.

Because each track on each surface is parallel to all other tracks with the same number, the tracks are logically combined into a series of nested **cylinders**. The number of tracks and the number of cylinders are the same, so disk manufacturers describe their disks in part by how many cylinders they have.

In addition to the tracks, each platter surface is divided into wedges radiating from the center, thereby dividing each track into units called **sectors**. Sectors are the smallest physical division on a hard disk; by de facto industry standard, sectors are 512 bytes in size. Figure 6-1 shows the disk-level logical divisions of a disk's storage space.

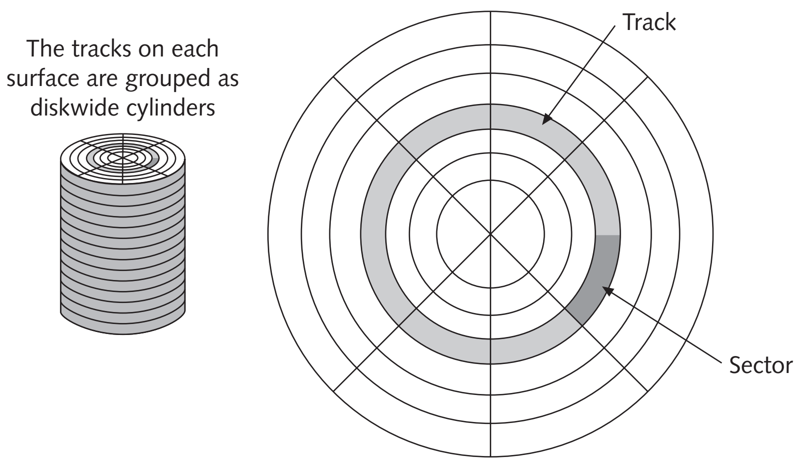

Figure 6-1 Physical disk geometry

The Partition Table

Disk space isn't readable on its own. To make the disk usable, you must logically organize its space into divisions called **partitions**. Each disk has at least one partition and, under normal circumstances, as many as four.

Note: Windows 2000 includes a type of disk format called dynamic storage that allows you to have more than four partitions on a disk. The later section on volume types discusses this format in more detail.

The first sector on a hard disk stores information that tells the computer how to read that disk. The **partition table** found in this sector is a 64-byte record of each partition (logical division) of the space on a hard disk. Each entry in the partition table is 16 bytes long, so the table may describe a maximum of 4 partitions (64/16 = 4). The entries for each partition include the following information:

- The partition's boot status—bootable or not bootable (8 bits)
- The surface where the partition starts and ends on the disk (8 bits)
- The cylinder where the partition starts and ends on the disk (10 bits)
- The location of the sectors where the partition starts and ends on the disk (6 bits)
- The number of sectors in the partition (32 bits)
- The file system used to format (logically organize) that partition (8 bits)
- If the partition is part of a multidisk volume, the other disks with which the partition works

The size of each field limits the size of the disk that the computer's BIOS (basic input/output system) can "see" without the help of an operating system such as Windows 2000. For example, only 8 bits are available to identify the surface where a partition starts. The entry is written in binary notation, so it can name as many as 2^8 (256) surfaces. No problem—you can just add more cylinders to those surfaces, right? Well, no. Because the entry for the starting and ending cylinders is only 10 bits long, the partition table can identify only disks with a maximum of 1024 (2^10) cylinders. And each cylinder can have only 64 (2^6) sectors in it, because the entry for sectors has only 6 bits. (Actually, it's worse than that, because sectors start numbering with 1, not 0 like most computing things do. Thus you have only 63 sectors per cylinder.) You can calculate the total capacity of a disk with the following formula:

```
Disk Capacity = sector size x sectors per track x cylinders
x heads
```

Therefore, the theoretical maximum size of a disk that the BIOS can see is 512 bytes × 63 × 1024 × 256, or 8,455,716,864—approximately 7.8 GB.

If you're looking in confusion at the 10 GB disk inside your computer and wondering how Windows 2000 reads it, fear not. Addressing disks larger than this theoretical limit is a job carried out by the operating system. Once Windows 2000 starts running, it controls the hard

disk controller without help from the BIOS (meaning that it's no longer dependent on the BIOS's ability to recognize hard disks). During its operation, Windows 2000 ignores the sections of the partition table that refer to absolute sector and cylinder numbers (which are only 6-bit and 10-bit entries), instead referring to the sections that note the relative sector numbers and the number of sectors (both 32-bit entries). Using these entries allows Windows 2000 to "see" hard disks with 2^32 sectors, or, using the standard 512-byte sector, 2 terabytes.

The Master Boot Record

The partition table is part of a larger structure called the **master boot record (MBR)**. On x86 systems, the MBR contains the partition table for the disk and a small amount of code that can read the partition table into memory and find the system (bootable) partition for that disk if one exists. The MBR then finds the first sector of that system partition on the disk and loads an image of its starting sector—its **boot sector**—into memory. The boot sector finds the files needed to run the operating system controlling that system partition (in the case of Windows 2000, it starts with a file called Ntldr) and loads those files into memory. Thus, the MBR is not operating system-specific, but the boot sector is.

If the disk is not bootable (that is, if it has no operating system installed on it), then the partition table is just loaded into memory where the rest of the computer can access it. (Data on disk is essentially invisible. Only data in memory is usable by any part of the computer or operating system.)

> Note: If the MBR or a boot sector isn't working or is missing, the computer cannot identify the system partition and the disk won't boot. A class of viruses called boot viruses target the MBR, because it's a sitting duck. Wipe out the first sector on the hard disk, and you've killed it. To prevent this event from happening, you should use a disk utility to copy the contents of the MBR so that you can restore it if necessary. The Windows 2000 Recovery Console includes a tool called FIXMBR that can create a new MBR on a hard disk, as long as the system disk is bootable. It also contains a tool called FIXBOOT that you can use to replace the boot sector on a partition. Chapter 14 discusses these tools in more detail.

Logical Disk Divisions

Earlier in this chapter, you saw the setup that the partition table uses to identify the system partition on a disk. Although you could use this same system to find files stored on a disk—and early operating systems did—using this technique would be time-consuming and tricky to set up and even trickier to keep updated.

If it's not clear why this technique is so challenging, consider this example: You are working on a spreadsheet. In your no-file-system environment, you have to know which cylinder holds the file, what surface it starts on, what sector it starts on, and how many sectors contain part of the file. Enter all that information, and you can load the spreadsheet. But what happens if you add to the spreadsheet? Even if it's only 512 bytes larger than it used to be, then you must edit the location data to take into account all sectors used by the file. Partition tables work because they deal with small amounts of data that can be updated easily, describing only four

elements on the disk. Your computer probably has thousands of files on it. Another problem with this kind of organization is that it's difficult to tell which clusters are used and which are not. Overwriting data can be prevented, but unused space might go to waste because you don't know that it's there.

Rather than attempt to create partition-table-like entries for files, operating systems use **file systems** to organize data on a disk. When you format a partition, you're running a program that evaluates the size of the partition, then, based on that size, groups sectors together into logical units called **clusters**, which are the smallest storage units available to the file system (see Figure 6-2). The number of sectors in a cluster depends both on the size of the partition and on the type of file system being used. Windows 2000 supports three file systems: **FAT (file allocation table)**, **FAT32**, and **NTFS (New Technology File System)**. **FAT16** was designed for backward compatibility with other operating systems; FAT32 is compatible with Windows 98 and more space-efficient than FAT16; and NTFS is designed for Windows NT/2000 and includes advanced features that make it more secure and more flexible than the other two file systems. The details of these file systems and how they organize disk space are discussed later in this chapter.

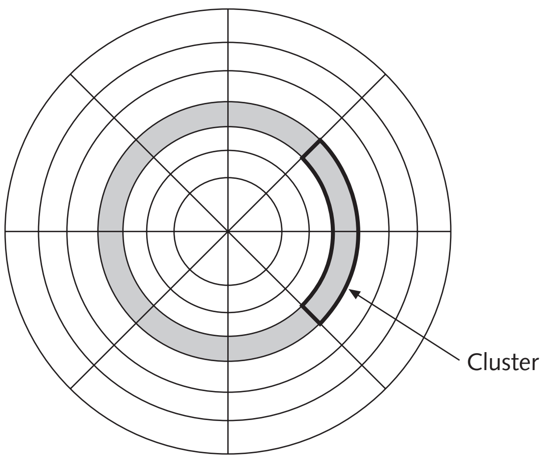

Figure 6-2 When you format a partition, its sectors are logically grouped into clusters

Volume Types

Clusters represent the smallest unit of disk space intelligible to Windows 2000. Those clusters may, in turn, be part of one of several kinds of **volumes** (another term for partitions). The kind of volume depends on the type of disk storage used on those disks. Windows 2000 accommodates two kinds of data storage: basic and dynamic. **Basic storage** supports the partition-oriented disk organization that has been discussed up to this point and is intelligible to all operating systems. **Dynamic storage** supports multidisk volumes. Because dynamic storage is unique to Windows 2000, volumes on dynamic disks—and the dynamic disks themselves—are invisible to any operating systems other than Windows 2000 that are installed on the same computer.

Basic Storage Types A partition is a portion of a hard disk set up to act like a separate physical hard disk. Two kinds of partitions exist: primary and extended. A **primary partition** is a portion of a physical hard disk that the operating system (such as Windows 2000) marks as bootable and assigns a drive letter. Windows 2000 supports a maximum of four partitions per disk, so you could have four operating systems installed on the same physical disk. Only one partition at a time is marked as being active—the one from which you booted. You cannot subdivide primary partitions; once you create them to be a certain size, they will remain that size until they are deleted.

If you want to subdivide a partition, you'll need an **extended partition**. An extended partition can't hold data; it's just a logical division of **unallocated space** (unpartitioned space) on the disk. Once you create an extended partition, you can divide the **free space** in it into **logical drives**, which have drive letters like primary partitions and can hold data once they're formatted. You can have one extended partition on each disk, but may divide that partition into as many logical drives as you like.

Dynamic Storage Types Any volumes that are extended over multiple disks, or that have the capability of becoming so, must be created on dynamic disks. Dynamic disks support several different kinds of storage—some fault-tolerant, and some not.

Note

Chapter 13, "Fault Tolerance," discusses the use of fault-tolerant volumes in detail.

The simplest kinds of dynamic volumes are **simple volumes**. Simple volumes are like primary partitions; they're created from unallocated space, are located on a single disk, are identified by a drive letter, and can be made bootable. Windows 2000, for example, can be installed on a simple volume. **Spanned volumes** are volumes that extend over more than one physical disk, but otherwise work like simple volumes. You can extend simple volumes to become spanned volumes.

Flexibility is the driving force behind simple and spanned volumes. Volume sets are not limited to one physical disk, so with a spanned volume, you can make quite a large volume set out of many small pieces of unallocated space. You can therefore use unallocated space more efficiently than you could if limited to a single disk. It's much easier to fit 300 MB of data into a 350 MB volume set than it is to fit it into two 100 MB logical drives and one 150 MB logical drive. Also, the data will be easier to find once you do fit it in, as it will be stored under one drive letter, not three. Because the data is accessible under a single drive letter rather than many drive letters, it is easier for the system to access the information.

You can increase the size of a volume set by extending it if a physical disk contains more unallocated space, but you cannot make a simple or spanned volume smaller. If you must, then you'll need to delete the volume and create a new one.

Another type of dynamic storage is the stripe set. A **stripe set** is a volume that extends across two or more physical disks. Unlike volumes, which may include areas of unallocated space

of any size on the disks, all areas of a stripe set must be the same size on each disk, because of the way in which data are written to a stripe set. When data are written to a volume, a partition or logical drive files are put into the first available clusters, filling up the area from front to back, as it were. Even spanned volumes use only one disk at a time—the second disk is there to handle the overflow when the first one fills up.

In a stripe set, data are written to the areas of the stripe set on each disk in a series of stripes. In other words, not all of your data ends up in one place. Even if one of the areas in the set has enough room for an entire file, the data won't all be written there. The reason for this choice is the attempt to improve read and write times for the volume. Whenever you write data to a disk, a device on the disk called a disk controller is responsible for the process. Using more than one disk means that you can use more than one disk controller, with the result that you reduce disk access time. In other words, all other things being equal, it's faster to read and write data to a stripe set than it is to a primary partition or spanned volume.

It's important to note that volume sets and stripe sets do not protect your data; they merely let you group available drive space efficiently or reduce disk access time. If a hard disk used in a volume set or stripe set stops working, all of the data on it becomes inaccessible, even if the other hard disks in the set are fine. It's true that the more disks you have, the less likely it is that all of them will fail at the same time. The flip side is that the more disks you have, the more likely it is that *one* of them will fail at any given time. In a situation in which one disk failure brings down the entire system, having more disks actually increases your vulnerability to hardware failures. For this reason, it's vital that the data on volume sets and stripe sets be backed up regularly.

Dynamic storage does support two types of **fault-tolerant volumes** (volumes that are arranged in such a way that even if one disk goes bad, your data will still be recoverable): disk mirroring and RAID 5 volumes.

Disk mirroring helps protect your data by storing a copy of it on an identically sized area of free space on another disk—thus a volume of 500 MB needs a mirror of 500 MB. The original and the copy are collectively known as a **mirror set**. If anything happens to one disk in the mirror set, you still have an identical copy of its data on the other disk.

The trouble with mirror sets is that they're not very space-efficient, because every piece of data that you record has an identical twin on the other half of the mirror set. You need exactly twice as much storage space as you have data. Therefore, Windows 2000 supports **stripe sets with parity**, also known as **RAID 5 volumes**. RAID 5 volumes work like the stripe sets mentioned previously, except that, in addition to writing the data in stripes to the disks in the stripe set, the disk controller also writes parity information evenly across the disks in the set. If one disk in the set dies, then the data contained on that disk can be regenerated from the parity information on the other disks in the stripe set. RAID 5 volumes are a little slower when it comes to writing data than other volumes, because of all the parity information that must be written, but they are a more space-efficient option than disk mirroring. Parity information takes up $1/n$ of the total disk space, where n is the number of disks in the set (3–32). Thus, the more disks in the RAID 5 volume, the more space-efficient it is. As shown in Figure 6-3, the Windows 2000 Disk Management tool displays all existing types of volumes and lists the redundancy factor for each.

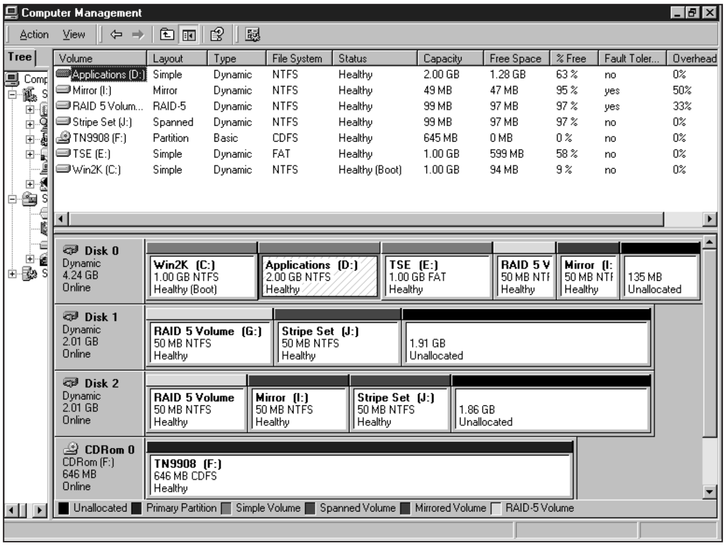

Figure 6-3 Redundant volumes don't have as much room to store data as other volumes do, but are less vulnerable to data loss

Note

RAID 5 volumes get their name from fault-tolerance terminology. A redundant array of independent disks (RAID) is a methodology of keeping data secure by storing the information on multiple disks such that if one disk fails, the data will still be available. Mirror sets are RAID 1 volumes. Windows 2000 calls disk striping with parity "RAID 5" volumes to avoid confusion with the disk striping without parity.

Identifying Partitions and Volumes For you to store data on a drive/volume/partition, that drive/volume/partition must have some identifier so that you can get its attention. Traditionally, in Windows operating systems, this name has been a drive letter—the boot drive, for example, is normally named C:. This method is simple and has the advantage of providing a really short way of leaping to that partition. The disadvantage, of course, is that as long as Windows 2000 insists on using the Roman alphabet, you're limited to a total of 26 letters for all local drives and mapped network connections. If you think you might need more than 26 total connections, you're better off mounting the partition to an empty folder on an NTFS volume.

The basic idea behind **mounting a partition** to a folder is that you redirect all read and write requests sent to that folder to the partition instead. Mount a new partition to G:\My Work Stuff, and every file I/O request you send to G:\My Work stuff will be rerouted to the new partition, even if the real drive G: is located on an entirely different physical disk. You can mount a volume to as many paths as you like; alternatively, you can mount it to both paths and a drive letter to let people access the same storage space from different paths. The only restrictions are that the folders must be empty at the time of mounting and not mapped to any other volumes, and they must be on NTFS volumes on the local computer.

Understanding Windows 2000 File Systems

Whichever kinds of volumes you choose to make, you must format these volumes with a file system before you can store data on them. The role of the file system is to organize the sectors in the disk space so that the instructions to the disk controller, telling it how to read and write data, can be as explicit as possible. As stated earlier, Windows 2000 supports three file systems: FAT16, FAT32, and NTFS. You can distinguish these file systems in three main ways:

- The method used to organize disk space
- The cluster size
- Special features available

FAT16

The FAT file systems were the first Microsoft-compatible file systems. They're simple, not meant for very large disks (when FAT was invented, very large disks didn't exist), and have only a few simple file attributes.

FAT16 and FAT32 use a catalog called the file allocation table to keep track of how clusters are used on the volume. Each cluster is identified as having one of the following states:

- Unused
- Cluster in use by a file
- Bad cluster
- Last cluster in a file

Each folder has a 32-byte entry in the FAT naming the files it holds. A **root folder**—which you see as the root directory on a partition—contains an entry for each folder in the volume. The only difference between the root folder and other folders is that the root folder is at a specified location and has a fixed size of 512 entries (for a hard disk; the number of entries on a floppy disk depends on the size of the disk).

A folder's entry contains the following information:

- Filename
- Attribute byte (8 bits)
- Create time (24 bits)
- Create date (16 bits)
- Last access date (16 bits)
- Last modified time (16 bits)
- Last modified date (16 bits)
- Starting cluster number in the file allocation table (16 bits)
- File size (32 bits)

FAT works like this: Each cluster in the partition has an address. When a file is stored on the volume, it goes into the first available cluster. If the file's too big to fit entirely into a single cluster, then the remaining data go into the next available cluster on the volume and the first cluster gets a pointer to the next cluster holding the file's data. This process continues until the file is completely stored, at which point the final cluster gets an End Of File (EOF) marker. The FAT, stored at the beginning of the volume, records which clusters are chain-linked together and which files are found where.

All operating systems that can read FAT volumes can read the information in the folder. Windows 2000 can store additional time stamps in a FAT folder entry, showing when the file was created or last accessed. These time stamps are used mainly by Portable Operating System Interface for UNIX (POSIX) applications—which is to say, not often.

Because all entries in a folder are the same size, the attribute byte for each entry in a folder describes what kind of entry it is. For example, one bit indicates that the entry is for a subfolder and another bit marks the entry as a volume label. Typically, only the operating system controls the settings of these bits. The attribute byte includes four bits that can be turned on or off by the user—archive file, system file, hidden file, and read-only file. The **archive attribute** marks a file for backup. The **system attribute** marks the file as a system file. The **hidden attribute** hides a file when turned on. The **read-only attribute** makes a file readable but prevents it from being changed.

The process of reading data from a FAT volume goes something like this:

1. An application attempts to open a file and calls upon the operating system to get the file. This request is passed to the I/O system in the Windows 2000 executive, and from there to the file system driver.

Note

If you're not sure how these parts of the operating system work together, review Chapter 2.

2. The file system driver inspects the FAT at the top of the volume to see whether the file exists.
3. The file system driver notes the location of the first cluster of data as listed in the FAT and retrieves the data.
4. Unless the file is entirely contained within one cluster, the FAT will show a pointer to the next cluster holding data for that file.
5. Step 4 is repeated until the driver finds a data cluster with an EOF notice signaling the end of the file data. The data are then passed to the application that requested the information.

Reading data in this way is a simple process for small volumes and requires little overhead. On large and very fragmented volumes, however, the FAT method of organization can be very slow. FAT also has another problem—it's not very efficient for organizing storage into clusters.

Each cluster in a FAT16 volume has a 16-bit address—hence the name FAT16. Having 16 bits in a binary number means that a FAT16 volume can hold a maximum of 2^16 (65,536) clusters. This limited cluster size means that big FAT volumes have to use big clusters so as to fit in all of the sectors. Table 6-1 reveals how cluster size increases with partition size in a FAT16 volume.

Table 6-1 Cluster sizes in various sizes of FAT volumes in Windows 2000

Volume Size	Number of Sectors/Cluster	Cluster Size
0 MB–15 MB*	8	4 KB
16 MB–127 MB	4	2 KB
128 MB–255 MB	8	4 KB
256 MB–511 MB	16	8 KB
512 MB–1023 MB	32	16 KB
1024 MB–2048 MB	64	32 KB
2048 MB–4096 MB	128	64 KB
*12-bit FAT.		

As, you can see, really big FAT volumes are so large as to be impractical for storing lots of data. Each cluster can hold only one file's data. Thus, unless you will regularly store only very large files, FAT volumes larger than 2 GB aren't very efficient. You can make clusters smaller by making the volumes smaller, but this option isn't a really great solution, either. To use clusters more efficiently, you need either FAT32 or NTFS.

FAT32

FAT32 is organized essentially the same way as FAT16. The main difference between the two file systems is that FAT32 uses 32-bit addresses for each cluster in a volume, so it can address more clusters within a single partition. As a result, FAT32 clusters can be smaller than FAT16 ones for the same volume—only 4 KB for all volumes from 512 MB to 8 GB.

Although FAT32 in Windows 95-98 supports compression, the version provided with Windows 2000 does not. If you want to compress files, you need to use NTFS.

NTFS

NTFS differs from the FAT file systems in several ways. First, it organizes disk space and data differently. Second, it uses different cluster sizes. Third, it supports many more attributes, providing native support for encryption, file compression, **disk quotas**, and file and folder compression.

Directory Organization

The flat-file database structure that FAT file systems use to organize data is compact and works well on small volumes, but can be slow on larger ones. To speed things up, NTFS uses a treelike hierarchical structure. A catalog located at the beginning of the volume, called the

master file table (MFT), maintains a 2 KB entry for each file in the volume. The MFT contains both entire files (if small enough) and pointers to the locations of those files.

All data files stored on NTFS volumes have the following attributes, arranged as shown in Figure 6-4:

- Header (H)
- Standard information (SI)
- File name (FN)
- Data
- Security descriptor (SD)

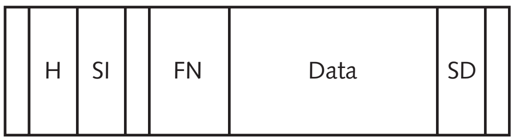

Figure 6-4 A simple NTFS file

The file may also include one or more of the system data attributes listed in Table 6-2.

Table 6-2 NTFS file attributes

Attribute	Description	Must Be Resident in Master File Table?
Attribute	Lists the attributes that did not fit into the MFT and their location in the NTFS volume structure.	Yes
Bitmap	Maps the sectors in use on the volume.	No
Data	Contains file data and the file size. A file may have more than one data attribute.	No
Filename	Defines the long and short filenames of the file and the file number of the folder containing the file. If a file appears in more than one folder, there will be a corresponding number of parent directory entries.	Yes
Index allocation	If the index becomes too large to be resident in the MFT, this nonresident attribute can hold the rest of the index.	No
Index root	Used to index files in folders by a given attribute, such as their names or size.	Yes
Object ID	Used to find files linked together in data streams. Files not linked do not have object identifiers.	No
Reparse point	Used to redirect NTFS when reading a path, such as when reading from or writing to a mounted volume.	No

Table 6-2 NTFS file attributes (continued)

Attribute	Description	Must Be Resident in Master File Table?
Security descriptor	Contains the security information for the file, defining who's got what kind of access to it. Also contains an audit field that can hold audit settings (defining which security-related activities associated with this file will be recorded).	No
Standard information attributes	Contain the information that you see when you view a file's property sheet: file creation date, last modification date for file data, last modification date for standard file attributes (read-only, hidden, archive, and system), and the like.	Yes
Volume information	Defines the version of NTFS used.	No
Volume name	Contains the volume label.	No

How does NTFS decide which attributes go into the MFT? As you can see from Table 6-2, some attributes must be **resident**, or stored in the MFT. The arrangement of attributes that don't fit in the MFT depends on just how many attributes are associated with the file.

Files with data attributes of less than 1500 bytes may be resident in their 2 KB entry in the MFT, because that space is generally adequate to store all attributes of such a small file. You can't be sure that small files will indeed be in the MFT (if they have more attributes than expected, they may not fit), but at that level it's possible.

Somewhat larger files won't fit in the MFT. Instead, their 2 KB record contains the attributes that must be resident and a data attribute that points to the true location of the data in the NTFS catalog. This pointer is the virtual cluster number (VCN) for the first cluster of each group of clusters that hold the data, as shown in Figure 6-5.

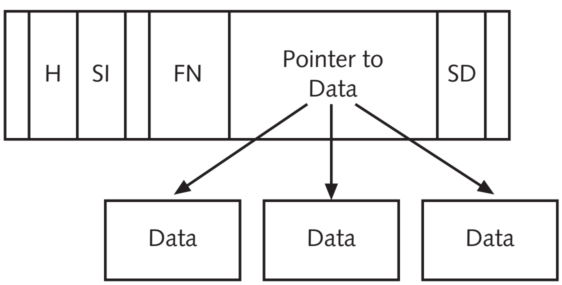

Figure 6-5 Files with data attributes too large to fit in the MFT have a VCN that points to the data's location in the volume

As the entry in the MFT is only 2 KB for each file, the data attribute might not be the only one squeezed out. In that case, very large files with lots of attributes may require a more cascaded structure in the NTFS catalog, as shown in Figure 6-6.

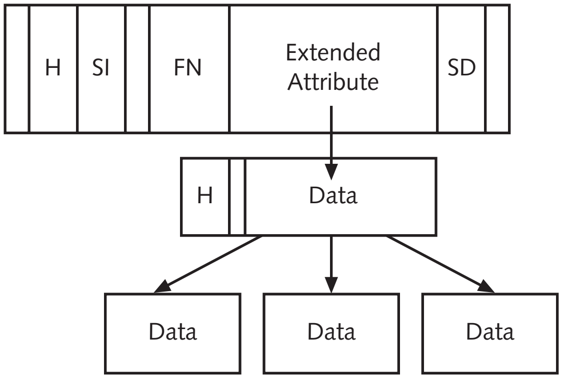

Figure 6-6 The largest files may require several data pointers to point to all their attributes

This tree-shaped database ensures that no file can ever be too large to fit within the NTFS structure. The more complex the file, the longer it will take to retrieve it, but the file will always fit.

Files aren't the only objects that get an entry in the MFT. The folders containing files also need entries, so that NTFS knows which folder contains the files. NTFS organizes folder objects in the same way as it organizes file objects, with one exception: folder objects can contain other objects (files and folders), whereas file objects cannot. The important parts of a directory are its name and its **index**, its list of contained folder and file objects. The name must be resident in the MFT, and the index should be. If the index is too large to fit in the 2 KB permitted for the folder's attributes, then the rest of the index is stored in a nonresident index allocation attribute.

Thus, the process of finding a file under NTFS is as follows:

1. An application attempts to open a file and calls on the operating system to get the file. This request is passed down to the I/O system in the Windows 2000 executive, and from there to the file system driver.
2. The file system driver checks the MFT to see whether such a file exists. If the file is small, then it may be in the MFT. If it is not, the MFT will contain a pointer to the file's attributes.
3. Having found the pointer in the MFT, the file system driver uses the pointer to jump to the file's attributes on the disk. If the file is there, then the driver opens it. If it's just a pointer to more attributes, the driver keeps moving down the directory tree.

This file system can potentially work more quickly on very large volumes, as the scope of the search is limited. Rather than searching the entire MFT, it's necessary to search only the part that points to the requested filename.

Important System Files

Any volume formatted with NTFS contains the following files:

- The MFT records the location and attributes of each file and folder in the volume. For safekeeping, Windows 2000 maintains a backup copy of the MFT on each NTFS volume in case the original is damaged or destroyed.
- A volume file holds the volume name and version.
- An attribute definitions file specifies all system-defined and user-defined attributes in the volume.
- A cluster allocation bitmap maps the storage space on the volume, excluding the cluster at the beginning of the volume. This map helps NTFS find unused space on the volume when it needs to store new files.
- A boot file contains the data that a system volume needs to boot.
- A bad cluster file records the location of all bad clusters (damaged clusters that cannot hold data) on the volume.

Note

If NTFS finds new bad clusters when attempting to write to disk, it records the clusters' locations and tries again to write the data on good clusters, instead of showing the "Error Reading Disk. Abort/Retry/Ignore?" message that you may have seen when attempting to write to a FAT volume with bad clusters.

Note

These NTFS system files may be located anywhere on the volume, rather than having to appear at the beginning of the volume as the FAT does. This flexibility makes NTFS volumes less vulnerable to bad areas of the volume than FAT volumes.

Cluster Sizes

Because NTFS uses 32-bit addressing, it can be more efficient in cluster allocations than FAT16 can. Table 6-3 shows the cluster size relative to partition size on NTFS volumes.

Table 6-3 Cluster sizes in various sizes of NTFS volumes in Windows 2000

Volume Size	Number of Sectors/Cluster	Cluster Size
512 MB or less	1	512 bytes (0.5 KB)
513 MB–1024 MB	2	1 KB
1025 MB–2048 MB	4	2 KB
2048 MB or more	8	4 KB

New NTFS Attributes

NTFS in Windows 2000 includes some attributes not found in either NTFS in previous versions of Windows NT or FAT. These new features make Windows 2000 perform better with large files, keep track of changes to the volume structure, limit the amount of disk space that

users' files can occupy, and encrypt their data. Many of these features are explained elsewhere in this book, but the following sections examine the features supported only by the NTFS file system.

Multiple **data stream** attributes allow developers to manage data in different files as a single unit. This feature works a little like mounting—the data files are mounted to a location in the data stream and are therefore linked. Streams can share data between files, but protect file information (such as file size) from being corrupted between files.

Reparse points alter the way that NTFS resolves filenames. For example, under normal circumstances, when you open the file located at C:\WINNT\System32\Example.doc, NTFS finds the C: volume, then finds the boot partition, then the WINNT folder, the System32 folder, and finally the Example.doc file. A reparse point can exist at any point in that path, directing NTFS to search the path differently. For example, when you mount a volume to an NTFS path, you use reparse points to redirect writes sent from one disk to another disk.

NTFS uses the **change journal** to log all changes—additions, deletions, or edits—made to files in the volume. Several volume tools use the change journal to tell when they're supposed to do their jobs: replication managers that copy changed data to a replication partner, the remote storage service that automatically backs up unused data to a tape drive, and incremental backups (which back up only changed files), can all use the change journal.

NTFS now supports the **Encrypted File System (EFS)** to encrypt data on the disk, including temporary files. This encryption prevents anyone from reading data from an operating system other than Windows 2000—it's mostly meant for people using laptops or as protection for removable media.

You may recall from Chapter 4 that when data are read into memory, memory is allocated in a two-stage process: first a process reserves memory that it wants to use, then it allocates that reserved memory by backing it with space in the paging file in case the data need to be paged out of main memory. With very large files, this technique can consume a lot of space in the paging file. That's fine, but some files have a great deal of redundant or nonmeaningful data (for example, long strings of 0s) that takes up space in the paging file. **Sparse files** avoid this problem by marking certain files with an attribute that causes the I/O subsystem to allocate memory only to the file's meaningful data. When a sparse file is read, the allocated data are returned as stored and the nonallocated data are returned as strings of 0s. Basically, the sparse file attribute tells NTFS and the Virtual Memory Manager, "there are nine 0s here," instead of making the Virtual Memory Manager find space in which to put "000000000".

Disk quotas let you restrict the amount of data that users can store on NTFS volumes. If a user tries to write more data than is permitted on that volume, the user will receive an "Access Denied" error.

Finally, NTFS protects itself from volume corruption. Every time you change an NTFS volume (perhaps by adding a new folder), a **transaction log** records the change you wanted to make. When the change has been successfully made, the transaction log notes that the change has been committed. If the disk stops working in the middle of the change to the

volume structure, then when Windows 2000 comes up again, NTFS refers to the transaction log for each volume and implements only those changes that have been fully committed. Thus, some changes to the volume structure may be lost, but NTFS volumes are kept from being corrupted by unfinished volume changes.

Caution

The only catch to NTFS for Windows 2000 is that it really works only for Windows 2000. Users with Windows NT Server 4 with Service Pack 4 can read and write to Windows 2000 NTFS volumes, but they can't use any of the special attributes for the volumes if accessing them on the local computer. If you need to support multiple operating systems on the same disk, you must use FAT16, as the most widely supported file system.

Chapter Summary

- Booting the computer from a hard disk isn't the job of the operating system, but rather is the responsibility of a set of instructions in a computer's BIOS. For those who'd like to use that hard disk, however, the operating system is key. An operating system such as Windows 2000 provides instructions that remove the four-partition limit imposed by the partition table; it also supports file systems that make reading and writing data to the disk simple and secure.
- Windows 2000 supports three different operating systems: FAT16, designed for backward compatibility with other operating systems; FAT32, compatible with Windows 98 and more space-efficient than FAT16; and NTFS, designed for Windows NT/2000 and boasting advanced features that make it more secure and more flexible than the other two file systems. The file system you choose will affect how you can use the data storage on your computer—so choose carefully.

Key Terms

archive attribute — A simple attribute that identifies a file as having changed since the last full backup.

basic storage — A hard disk designed to support primary and extended partitions and logical drives. Any operating system can recognize disks set up to use basic storage.

boot sector — An area at the beginning of each partition that names the files to be loaded to run the operating system stored on that partition.

boot virus — Malicious software that targets the master boot record of a disk to make the disk unbootable. Until the advent of macro viruses, boot viruses were the most common virus type.

change journal — A list of all changes made to files in the volume. Some Windows 2000 functions, such as the remote storage service, can refer to the change journal to know when to do their jobs. The change journal is a more efficient way of looking for changes than browsing the volume looking for the desired difference.

cluster — A logical grouping of sectors, with the number of sectors per cluster depending on the size of the partition and the file system being used. A cluster is the smallest storage unit that Windows 2000 file systems can recognize.

cylinder — All of the parallel tracks on all surfaces. For example, Track 10 on all surfaces creates Cylinder 10 for the disk.

data stream — Chunks of data that may be associated with more than one file. Data streaming allows you to deal with several distinct pieces of data as one unit.

disk quotas — A method of preventing users from using more than a predetermined amount of space in a volume. When a user exceeds his or her quota, he or she will be denied write access to the volume until some files have been deleted to go below the quota.

dynamic storage — A new type of storage in Windows 2000 that designs disks to support multidisk volumes. Volumes on dynamic disks may be added, resized, and deleted without rebooting.

Encrypted File System (EFS) — The Windows 2000 native encryption service, which requires Windows 2000 to read files. Intended mainly for people with laptops and for removable storage that's vulnerable to theft.

extended partition — A disk partition on a basic disk that's designed to hold logical drives. Extended partitions can't hold any data on their own—they're just areas of free space in which you can create logical drives. A hard disk may hold one extended partition, but you can make as many logical drives within that partition as you like.

FAT (file allocation table) — A catalog at the beginning of a volume that notes each file and folder in the volume and lists the clusters in which each file is stored.

FAT16 — A file system first used with DOS and supported in Windows 2000 for compatibility reasons—only Windows 2000 can read NTFS volumes, so if you need to support dual-boot machines or write data to floppy disks, you need FAT. FAT16 uses a 16-bit addressing scheme for clusters and can support only fairly small volumes without wasting space from overlarge clusters, but it has little overhead.

FAT32 — A version of FAT that uses a 32-bit addressing scheme, so that it can address more clusters than FAT16.

fault-tolerant volume — Any volume designed to reduce the risk of data loss due to disk failure. Fault-tolerant volumes either keep a copy of data or maintain information from which that data may be regenerated.

file system — A method of logically organizing the physical disk space in a partition for use by the operating system. Different file systems catalog data differently and support different file attributes.

free space — An area of an extended partition not yet made into a logical drive.

head — The read-write mechanism in a disk. Each surface has its own head.

hidden attribute — A simple attribute that hides a file. If the hidden attribute is set, the file will not show up in a DIR listing of the folder's contents, or in Windows Explorer unless hidden files are visible.

index — A list of all files in a folder in an NTFS volume.

logical drive — A formattable division of an extended partition, created from an area of free space. An extended partition may hold as many logical drives as you like.

master boot record (MBR) — A file stored in the first sector of a hard disk. It contains the partition table and links to the boot sectors for all partitions.

master file table (MFT) — A file in each NTFS volume that contains a 2 KB entry for each file and folder in the volume. If the file plus all attributes (including the data attribute) is smaller than 2 KB, then it may be stored in the MFT itself; otherwise, the file's entry in the MFT contains a pointer to the rest of the file's attributes that wouldn't fit.

mirror set — A fault-tolerant volume that exists in two identical, linked volumes on two dynamic disks. When you write data to a mirror set, the information is written to both volumes so that if one disk fails, the data will be recoverable from the other volume.

mounting a partition — Logically linking a volume to an empty folder on another NTFS volume. It means that you can write data to the path on one volume and have that data actually stored on the mounted volume.

NTFS (New Technology File System) — The native file system for Windows NT that is extended in Windows 2000. NTFS has many advanced features that make it more efficient and faster on large drives, supports volume mounting, and offers other features such as disk compression, file quotas, and a native encryption system.

partition — A logical division of disk space. A disk must be partitioned, and the partitions formatted, before it can be used. Disks can have a maximum of four partitions without the help of an operating system.

partition table — A table stored in the first sector of a hard disk, noting the location and size of every partition on the disk and indicating whether those partitions are bootable.

platter — A magnetized metal disk within a hard disk—the actual storage medium.

primary partition — A disk partition on a basic disk that's designed to hold an operating system (although it doesn't have to do so—a primary partition might hold only data). One primary partition is marked active, meaning that the computer will boot from it. A disk may hold a maximum of four primary partitions. Primary partitions may not be subdivided.

RAID 5 volume — A fault-tolerant volume extending over 3–32 disks. It works like a stripe set, except that in addition to writing data in stripes across the disks in the volume, it also writes parity information for the volume. If one disk in the RAID 5 volume fails, then the data on that disk may be regenerated from the parity information on the other disks.

read-only attribute — A simple attribute that makes it impossible to edit a file.

reparse points — NTFS pointers that may be set into a file path to redirect the path from one volume to another. Reparse points make mounted volumes work.

resident — Attributes that are stored in the master file table instead of being pointed to are known as resident attributes. Some attributes are required to be resident.

root folder — The folder in the FAT that lists all folders in the volume and all files in the root directory. A root folder can contain a maximum of 512 entries.

sector — The smallest physical unit of storage on a hard disk.

simple volume — A volume on a dynamic disk that exists on a single disk. Simple volumes may be expanded on the same disk or made into spanned volumes that extend to another physical disk.

spanned volume — A volume that extends over two or more dynamic disks.

sparse files — Files marked with an attribute that says, "Only provide space in the paging file for the parts of this file that actually have data in them, instead of strings of 0s." The data have pointers to the places where the long strings of 0s can be, so that they can be filled in as necessary, but sparse files save room in the paging file and in memory by allocating only the storage that's actually needed.

stripe set — A volume that extends over two or more dynamic disks, but which reduces disk read and write times by writing data to all disks in stripes, instead of filling up the volume from back to front as normal volumes do.

stripe set with parity — See RAID 5 volume.

surface — The side of a disk platter. Each platter has two surfaces.

system attribute — A simple attribute that identifies a file as part of the operating system.

track — A concentric circle traced on the surface of a platter, used to physically divide storage space.

transaction log — A list of changes to the volume structure maintained by NTFS. When changes are complete, they're listed in the transaction log as being committed. If the disk stops working, when it restarts, NTFS rolls back the volume structure to its form at the last committed change. This technique prevents the volume structure from being corrupted by half-made changes.

unallocated space — An area of a physical disk that has not yet been partitioned.

volume — Another name for a partition—a logical division of physical disk space. Most often, volumes refer to areas on dynamic disks, whereas partitions refer to the division of basic disks.

REVIEW QUESTIONS

1. A hard disk has as many heads as it has cylinders. True or False?
2. The smallest physical storage unit on a hard disk is called a(n) ________________. By de facto standard, it's ________________ in size.
3. Which of the following describes the limit on how many partitions a basic disk may have? (Select all that apply.)
   a. Four primary partitions
   b. Four primary partitions and one extended partition
   c. Four partitions of any kind
   d. A maximum of four extended partitions
4. The ________________ is a 64-byte record of each logical division on the disk. It's stored in the first sector on the hard disk.
5. If an operating system must depend on the computer's BIOS to handle hard disk operations, the disk may be as large as ________________ in size.

6. Which of the following statements is true?
   a. Windows 2000 can read disks larger than those that the BIOS can read because it refers only to those sections of the partition table that note relative sector numbers and the number of sectors.
   b. Windows 2000 can read disks larger that those that the BIOS can read because it refers only to those sections of the partition table that note the absolute sector numbers and the cylinder numbers.
   c. Windows 2000 uses logical block addressing to read disks larger than 10 GB.
   d. None of the above.

7. Which of the following contains the code that finds the bootable partition for a hard disk?
   a. The partition table
   b. The master file table
   c. The master boot record
   d. The logical block
8. Briefly describe the role of the boot sector.
9. The boot sector and the MBR are required for the disk to be recognized and to boot. True or False?
10. The smallest logical unit of storage in a Windows 2000 file system is the ________________.
11. List the file systems that Windows 2000 supports and identify the one designed specifically for use with Windows 2000.
12. ________________ storage supports partition-oriented volumes and is visible to any operating system (although the operating system must be able to read the file system to read the drive). ________________ storage is designed for use with Windows 2000 only and is required for creating multidisk volumes.
13. How many sectors are in a cluster?
14. How many operating systems could you install on a basic disk?
    a. One
    b. Four
    c. As many as you have extended partitions
    d. One for each logical drive on the disk
15. Which of the following are fault-tolerant volume types?
    a. Spanned volumes
    b. Mirror sets
    c. Simple volumes
    d. Stripe sets
16. ________________ are used to improve disk throughput on dynamic disks.

17. How do the two types of fault-tolerant volumes supported by Windows 2000 differ in terms of the amount of space for storing data you get in the volume?
18. Name all the Windows 2000 file systems that support partition mounting.
19. How many clusters does it take to store a 1 MB file under FAT16? Under FAT32? Under NTFS?
20. Both FAT32 and NTFS support file and folder compression. True or False?
21. How large is each file's entry in the master file table?
22. Explain how FAT file systems view data differently from NTFS.
23. NTFS is the only Windows 2000 file system that stores security information with a file's data. True or False?
24. Which of the following features of NTFS is used to make partition mounting work?
    a. Sparse files
    b. Data streaming
    c. Reparse points
    d. Both a and c
25. The ________________ attribute in NTFS stores the list of files in a folder.

HANDS-ON PROJECTS

Project 6-1

When installing Windows 2000, you must create a partition on which to store the operating system. If you make the partition smaller than the entire disk space, you can create another partition (or several) on the disk.

Note: You must have unallocated space on your hard drive to complete this exercise.

To create a 100 MB primary partition on a basic disk:

1. Open the **Computer Management** tool in the **Administrative Tools** folder, and access the **Disk Management** tool (see Figure 6-7).

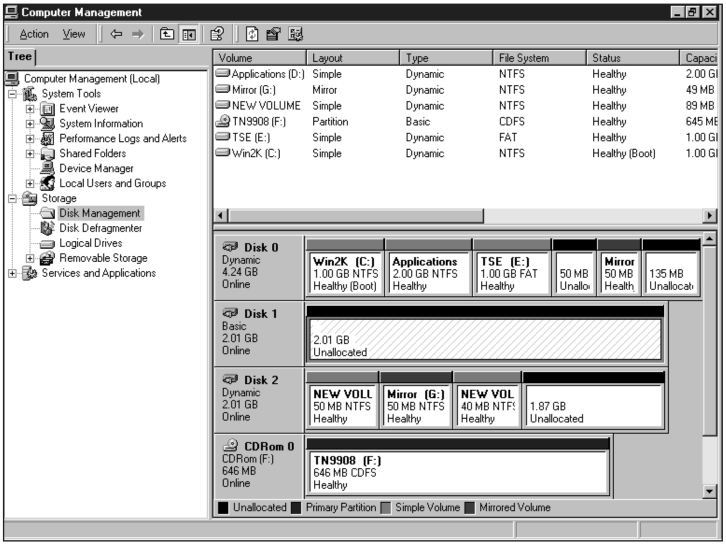

Figure 6-7 Use the Disk Management tool to create, extend, and delete volumes and partitions

2. Right-click on an area of unallocated space on a basic disk, and choose **Create Partition** from the resulting menu.
3. Click through the opening screen of the wizard. From the screen shown in Figure 6-8, choose a partition type. In this case, choose to create a **Primary partition**. Click **Next**.

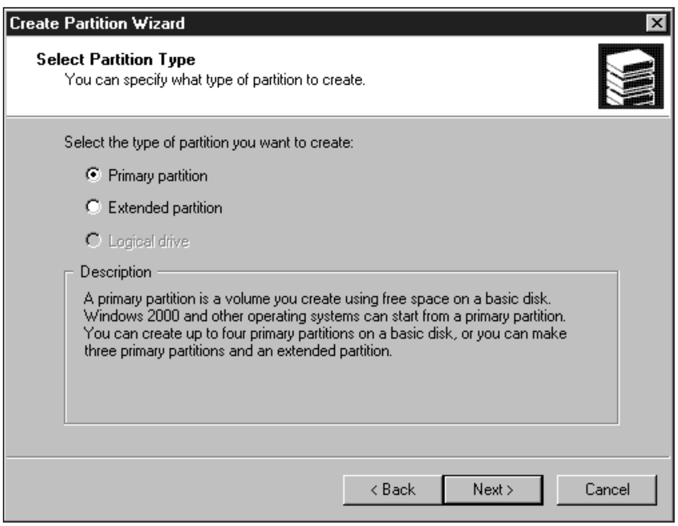

Figure 6-8 Pick a partition type from the options available

4. Make the new partition small enough to leave some room on the volume (see Figure 6-9). In this example, make it **100 MB**. Click **Next**.

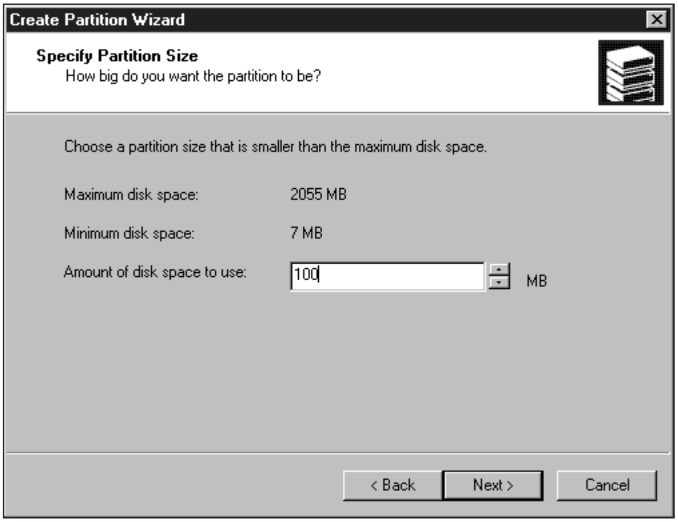

Figure 6-9 A partition doesn't have to use all of the available unallocated space

5. Choose a drive letter for the new primary partition, as shown in Figure 6-10. Name it **Drive X:**, and click **Next**.

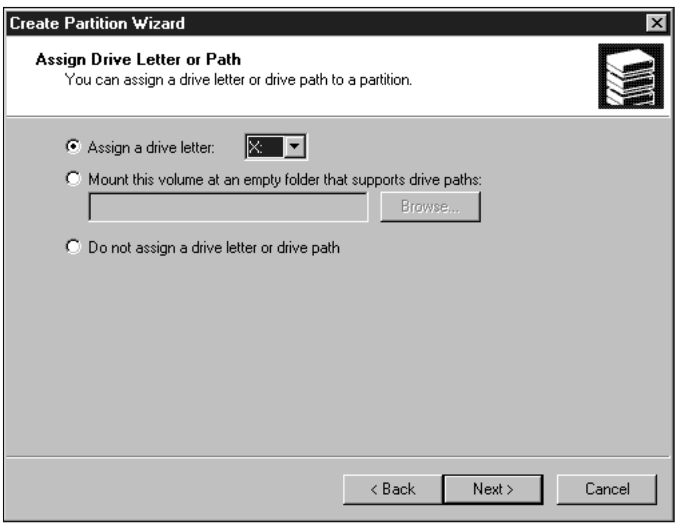

Figure 6-10 You can identify volumes and partitions by drive letter or by mounting them to a volume

6. Choose to format the new partition. For this example, choose **NTFS**, as shown in Figure 6-11. Click **Next**.

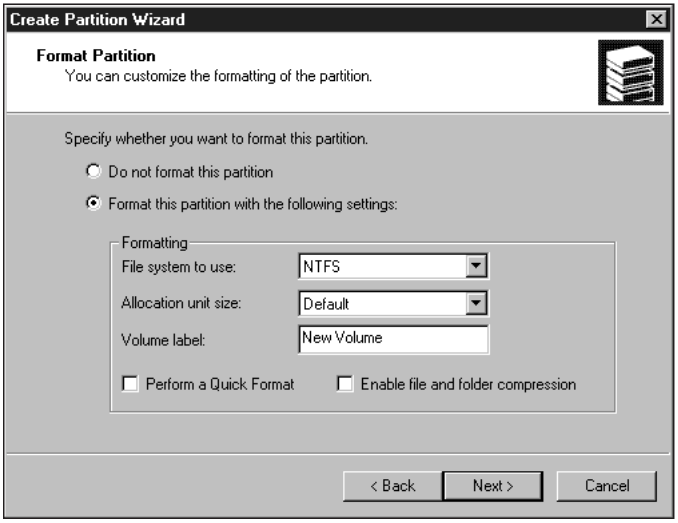

Figure 6-11 NTFS is the default file format for volumes created with the Disk Management tool

7. Review your choices in the final screen of the wizard, then click **Finish** to create the partition. Windows 2000 will take a moment to format the new partition, and then the partition will appear in the Disk Management tool as a segment with a dark blue stripe and labeled “X:”.

Project 6-2

You can create only four primary partitions on a basic disk. If you need more logical volumes, you must create an extended partition and then create logical disks from the free space within that extended partition.

To create an extended partition and then create logical disks from the free space within that extended partition:

1. Right-click on an area of unallocated space on a basic disk, and choose **Create Partition** from the resulting menu.
2. Click through the opening screen of the wizard, and choose a partition type. In this case, choose to create an **Extended partition** as shown in Figure 6-12. Click **Next**.

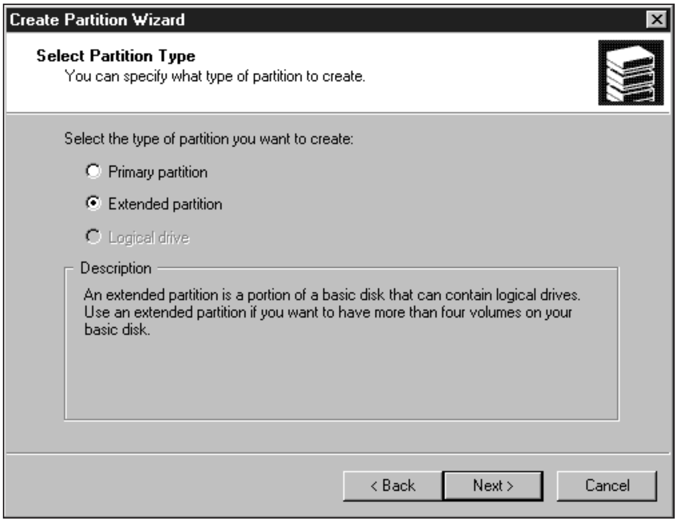

Figure 6-12 Create an extended partition so that you can create logical drives

3. Make the new partition small enough to leave some room on the volume (see Figure 6-13). In this example, make it **100 MB**. Click **Next**.

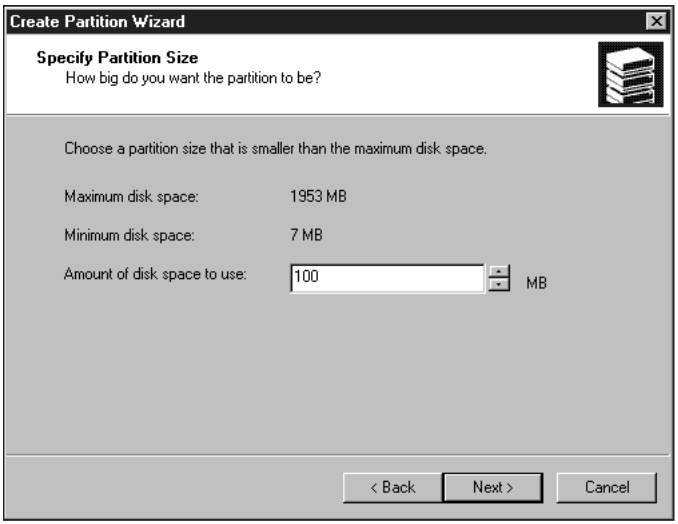

Figure 6-13 Make the extended partition large enough to hold the logical drives

4. In the final screen, review your choices and click the **Finish** button.

The new extended partition will be shown in the Disk Management tool with a thick, green line around it, and its space will have a bright green border and be marked Free Space.

5. Right-click on an area of free space within the extended partition, and choose **Create Logical Drive** from the resulting menu.

6. Click **Next** on the opening screen of the wizard, then choose to create a **Logical drive** (it will be the only option available; if it isn't, then you haven't chosen free space as your starting point). Click **Next**.
7. Pick a size for the logical drive. As shown in Figure 6-14, you can choose any size up to the size of the extended partition. Click **Next**.

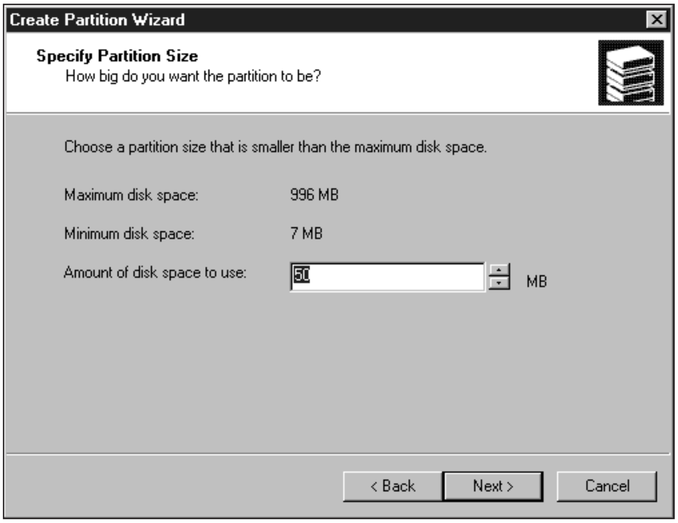

Figure 6-14 Choose a size for the logical drive

8. Assign a drive letter to the logical drive, as shown in Figure 6-15. For this project, choose **I**, and click **Next**.

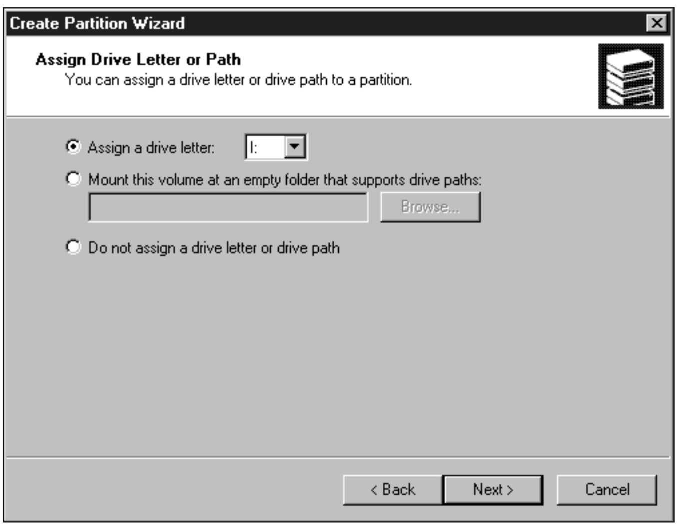

Figure 6-15 Identify the new logical drive with a drive letter

9. Choose to format the new logical drive with **FAT32**, as shown in Figure 6-16. Click **Next**.

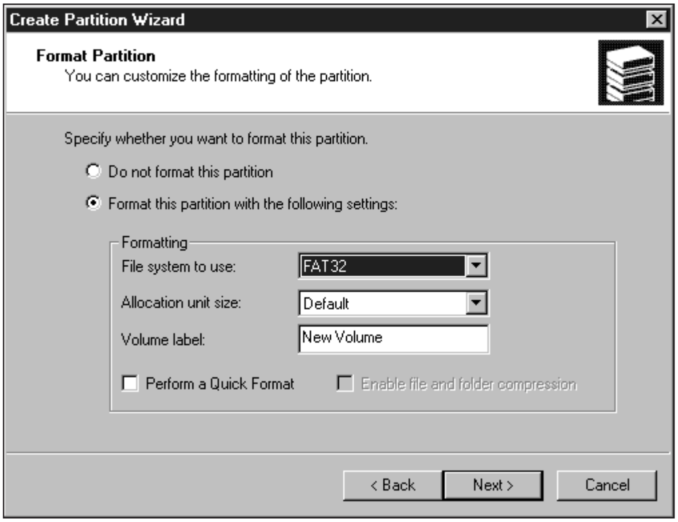

Figure 6-16 Format the new logical drive with FAT32

10. Review your choices in the final screen of the wizard, and click **Finish** to complete the new logical drive.

A new logical drive with a bright blue stripe will appear within the extended partition you created earlier.

Project 6-3

NTFS is the native Windows 2000 file format and supports many advanced features that the other two file systems do not. You don't have to reformat a disk if you want to stop using FAT16 or FAT32—just convert to NTFS.

To convert a FAT or FAT32 volume to NTFS:

1. Find a FAT or FAT32 volume and note its drive letter and volume name, if it has one. In this example, we'll convert **Drive I**.
2. Open the command prompt (available in the **Accessories** folder).
3. From the command prompt, type **convert i: /fs:ntfs**. Enter a volume name if prompted. You'll see output like that in Figure 6-17.

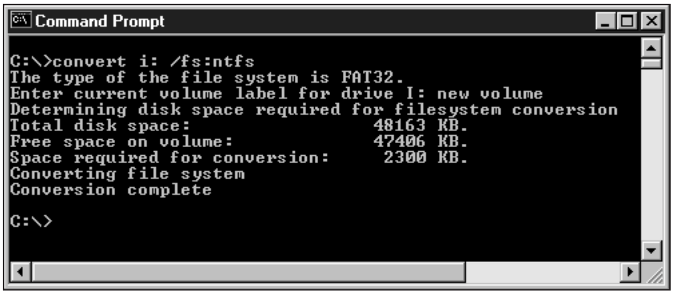

Figure 6-17 Converting a drive to NTFS

Tip: Drive I is now formatted with NTFS. If it doesn't appear in the Disk Management tool as NTFS, right-click the Disk Management folder in the left pane and choose Refresh to refresh the display.

Project 6-4

To create extensible or fault-tolerant volumes on hard disks, you'll need to first upgrade those disks to dynamic disks:

1. Right-click on the gray area of a basic disk (preferably not the one on which your operating system is installed, if you have a choice) and choose **Upgrade to Dynamic Disk** from the resulting menu.
2. In the dialog box that appears (see Figure 6-18), choose all basic disks that you want to upgrade. Click **OK**. You don't necessarily have to upgrade all disks.

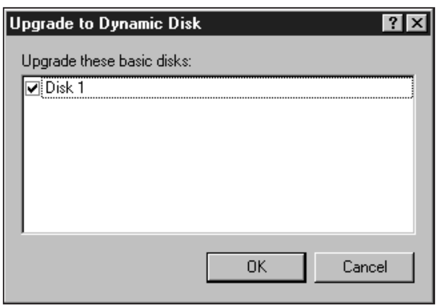

Figure 6-18 Choose disks to upgrade to dynamic disks

3. The Disk Management tool will display the disks you selected. Click **Upgrade**, then **Yes** to confirm the upgrade, and the disks you selected will be upgraded.

Project 6-5

Mounting volumes to drive paths offers several benefits: It permits you to add more space to volumes (such as partitions on basic disks) that couldn't otherwise be made larger; it allows you to make part of a non-fault-tolerant volume become fault-tolerant (if the volume you're mounting is fault-tolerant, that is); and it removes the 26-letter restriction on creating volumes.

To create a simple volume and mount it to a new folder on an NTFS volume:

1. Right-click on an area of unallocated space on the dynamic disk, and choose **Create Volume** from the resulting menu.
2. Click through the opening screen of the wizard. From the next screen, shown in Figure 6-19, choose to create a **Simple volume**. Click **Next**.

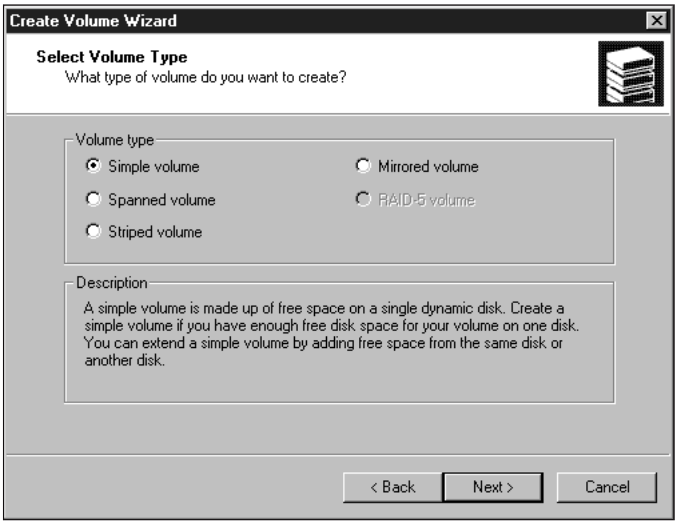

Figure 6-19 Dynamic disks have options different from those of basic disks

Note: The options available in this box will depend on the number of dynamic disks with unallocated space on them that are available. To create a stripe set or mirror set, you'd need two such disks. To create a RAID 5 volume, you'd need at least three such disks.

3. Choose a size for the simple volume, as shown in Figure 6-20.

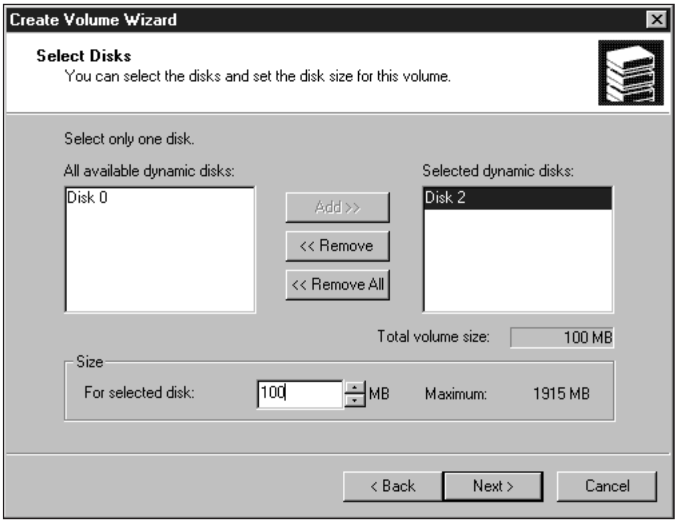

Figure 6-20 Size simple volumes just as you would a partition

4. In the next screen (shown in Figure 6-21), click the option that says to mount the volume and click the **Browse** button.

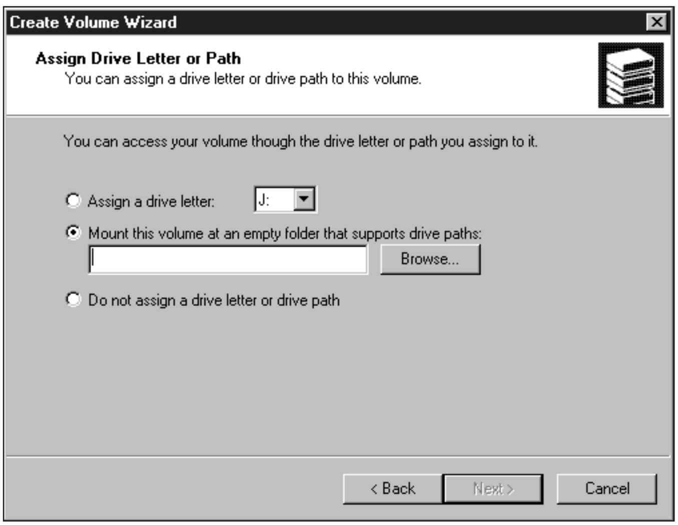

Figure 6-21 Choose to mount the volume, rather than assigning a drive letter

5. Browse for drive **X**:, the NTFS partition you created earlier, from the list shown in Figure 6-22.

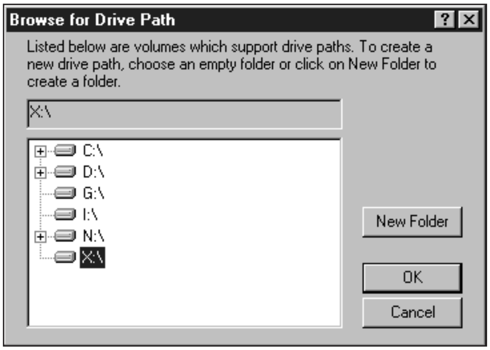

Figure 6-22 Choose drive X: from the list of disks that will support mounted volumes (NTFS volumes)

6. You'll need to create an empty path on drive X: on which to mount the new volume. Click the **New Folder** button, and type a name for the new volume, as shown in Figure 6-23. Click **OK** when you're done; you'll return to the screen where you chose to mount the new volume and the path you selected will be displayed. Click **Next**.

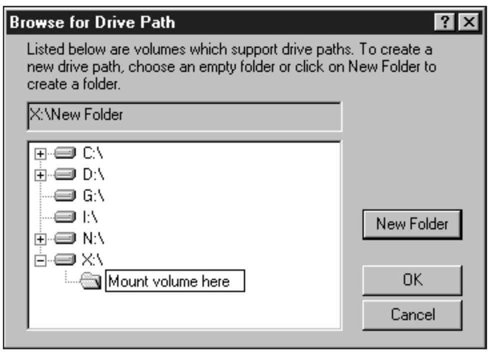

Figure 6-23 You must mount the drive to an empty volume

7. Format the new volume with **NTFS**, and click **Next**.
8. Review your choices in the final screen of the wizard, and click **Finish**. The Disk Management tool will take a moment to create and format the new volume. It will have a mustard-yellow stripe and will not show a path or a drive letter.

Project 6-6

One advantage of dynamic disks is that the NTFS simple or spanned volumes you create can be dynamically resized to make them larger if necessary.

To extend the simple volume that you created earlier to include an area of unallocated space on another dynamic disk:

1. Make sure that at least two dynamic disks with unallocated space are available, and right-click on the simple volume you created. Choose **Extend Volume** from the resulting menu.
2. Click through the opening screen of the wizard, and then choose the disk or disks onto which you want to extend the volume. As shown in Figure 6-24, you can extend the volume onto the same disk or another disk if another dynamic disk with unallocated space is available. Make sure that you choose a different dynamic disk from the one on which you originally created the volume.

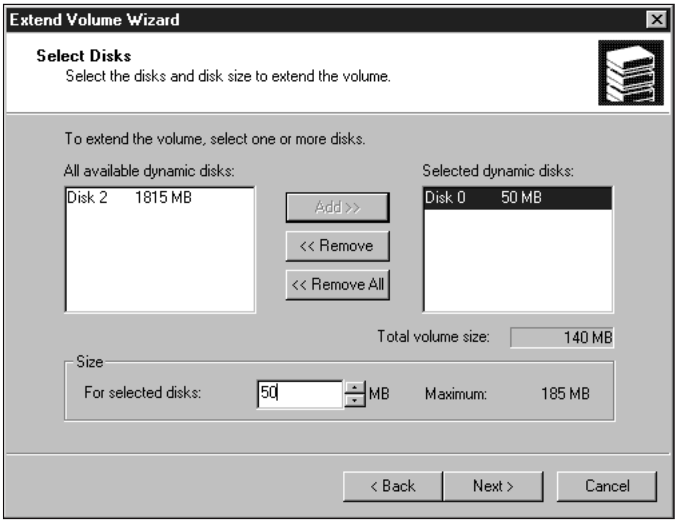

Figure 6-24 Find a dynamic disk with unallocated space available

3. In the same screen, choose the amount of unallocated space you want to add to the volume.
4. In the final screen of the wizard, review your choices and click **Finish**. Windows 2000 will extend the simple volume onto another disk and format the new area with NTFS. The simple volume is now a spanned volume, and has a purple stripe.

Case Projects

1. You're choosing file systems to use on a couple of computers. One computer is a file server with a 14 GB drive, which will have no other operating systems on it. The other computer is a desktop machine set up to dual-boot with Windows 2000 and Windows 98. Each computer has only one disk. Which file systems should—or could—you use on each computer? What's the basis of your decision?
2. Take the same computers from Case 1 and consider whether you would be well advised to use basic or dynamic storage with each. Would it make a difference if the computers had three disks each instead of one?
3. What would be the result of a virus destroying the master boot record on a hard disk? What would happen if the virus destroyed only the partition table?
4. You've created a terminal server with a complete set of applications and application tweaks that make the applications perform perfectly in a multiuser environment. How could you use the Windows 2000 RAID capabilities to duplicate this server?